AI Tech Boostcamp
Wrap-Up Report

팀 명	ConVinsight
기 간	2022.03.21 ~ 2022.04.07

About Project

00. Notion

https://yummy-angle-b95.notion.site/CV-05-Wrap-Up-Report-3b569e864aee4c3abe90a2a2e5c9b643

01. Synopsys

A. Subject

쓰레기 재활용 과정의 분리배출 검사를 위한 Object Detection Task

B. Synopsys

프로젝트명	재활용 품목 분류를 위한 Object Detection
기간	2022.03.21 ~ 2022.04.07
내용	우리는 많은 물건이 대량으로 생산되고, 소비되는 시대를 살고 있습니다. 하지만 이러한 문화는 '쓰레기 대란', '매립지 부족'과 같은 여러 사회 문제를 야기합니다. 분리수거는 이러한 환경 부담을 줄일 수 있는 방법 중 하나입니다. 따라서 우리는 사진에서 쓰레기를 Detection 하는 모델을 만들어 이러한 문제점을 해결해보고자 합니다. 만들어진 모델은 쓰레기장에 설치되어 정확한 분리수거를 돕거나, 어린아이들의 분리수거 교육 등에 사용될 수 있을 것입니다.

C. Tool & Env.

1. Ai Stages Project Server(GPU: Tesla V-100, OS: Ubuntu 18.04.5LTS)
2. GitHub
3. Notion
4. mmdetection

D. Structure

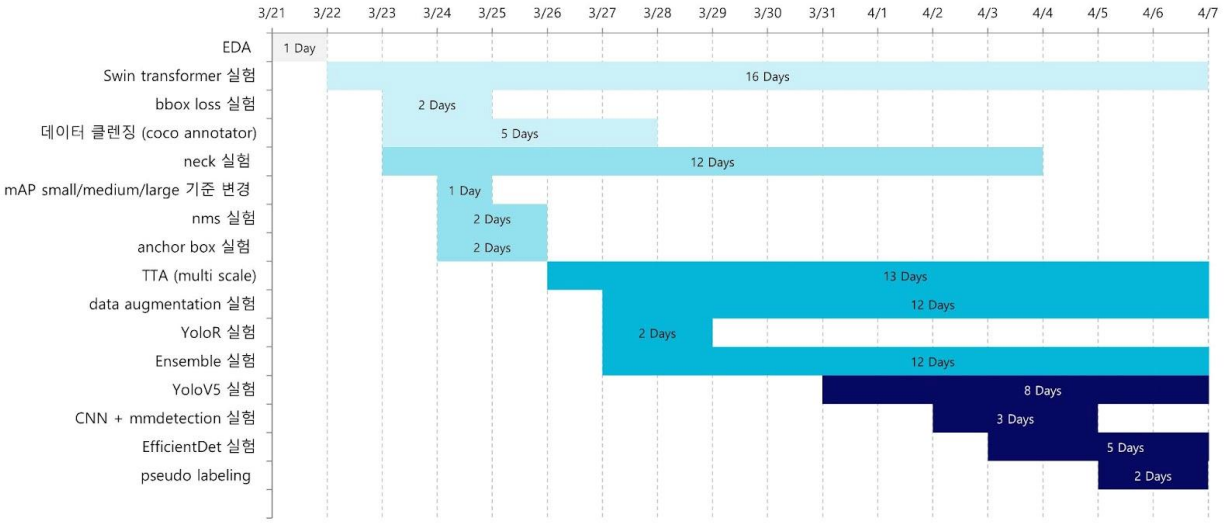
```
📂 detection/
|
├── 📂 baseline
|   |
|   ├── 📂 Swin_Transformer_Object_Detection
|   |   └── 📂 configs
|   |       └── 📂 p-stage
|   |           ├── 📂 __base__
|   |           |   ├── 📑 cascade_rcnn_swin_Base_fpn.py
|   |           |   └── 📑 cascade_rcnn_swin_Large_fpn.py
|   |           └── 📑 setup.py
|   ├── 📂 custom_configs
|   |   └── 📂 CNN
|   |       └── 📑 detectors_cascade_rcnn_resnext101_fpn.py
|   ├── 📂 efficientdet
|   |   └── 📂 effdet
|   |       ├── 📂 data
|   |       |   ├── 📑 dataset_config.py
|   |       |   └── 📑 transforms.py
|   |       └── 📑 train.py
|   ├── 📂 utils
|   |   ├── 📂 Compute_mean_std
|   |   ├── 📂 EDA
|   |   ├── 📂 EfficientDet_utils
|   |   ├── 📂 K-Fold
|   |   ├── 📂 inference
|   |   ├── 📂 multilabel_Kfolds
|   |   ├── 📂 oversampling
|   |   ├── 📂 pseudo_label
|   |   ├── 📑 csv2json.py
|   |   └── 📑 label_cleansing.py
|   ├── 📂 yolodata
|   ├── 📂 yolor
|   └── 📂 yolov5
|       ├── 📂 models
|       |   └── 📑 yolo.py
|       ├── 📂 utils
|       |   └── 📑 augmentations.py
|       ├── 📑 inference.py
|       └── 📑 train.sh
└── 📂 dataset
```

02. Team

A. Member

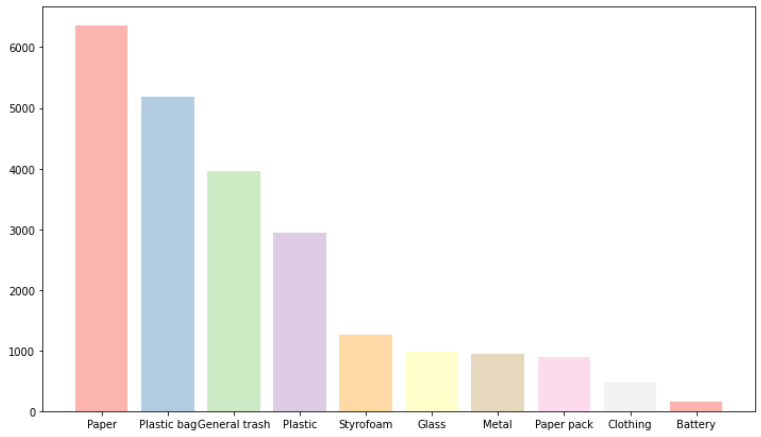
김나영 T3020	EDA, Data Cleansing, Model Experiments, Data Augmentation, Model Searching
신규범 T3116	EDA, Data Cleansing, Data Customize, Data Augmentation, Model Experiments, Ensemble
이정수 T3163	Data Cleansing, Hyperparameter Tuning, Model Experiments
이현홍 T3175	Data Cleansing, Model Searching, Data Augmentation
전수민 T3191	Data Cleansing, Data Augmentation, Model Experiments, Hyperparameter Tuning

03. Procedure



A. EDA

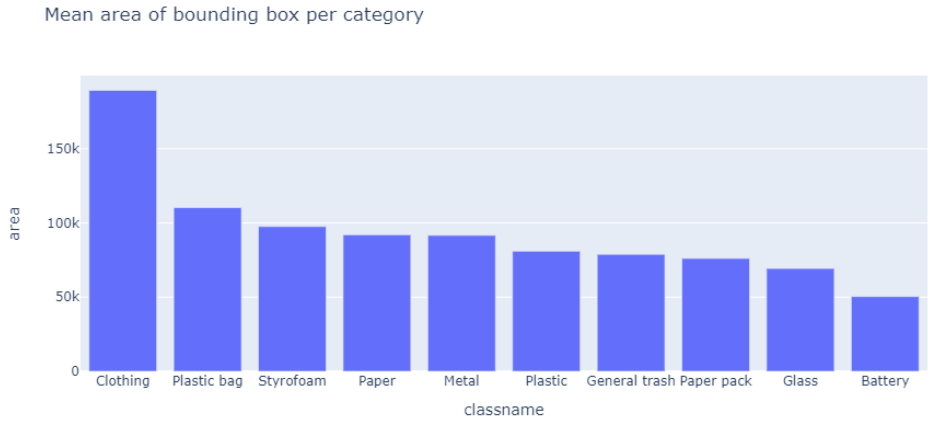
|Category별 boundingbox의 분포



General trash 0.284
Paper 0.515
Paper pack 0.580
Metal 0.456
Glass 0.615
Plastic 0.411
Styrofoam 0.589
Plastic bag 0.710
Battery 0.717
Clothing 0.480

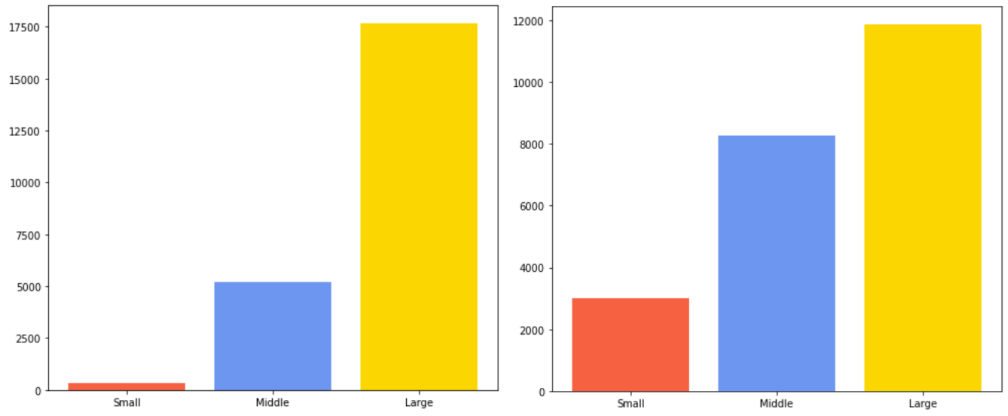
Class별 Bounding Box의 개수의 Imbalance 문제
- class별로 존재하는 bbox의 개수간 상이한 차이가 존재
- test용 모델을 학습 시켰을 때 class별 mAP와 bbox의 개수 사이에 특별한 관계는 보이지 않아 class imbalance는 모델 전체적인 성능과 크게 연관이 없을 것이라고 추측

|Category별 bounding box의 면적 분포



- battery는 다른 object들에 비해 실제 크기가 작음에도 불구하고 bounding box의 면적 차이가 적은 이유는, battery를 포함한 사진이 다른 사진에 비해 상대적으로 가까이에서 촬영한 경우가 많았기 때문

|bounding box 크기의 분포



(왼쪽) mAP_s, mAP_m, mAP_L의 기준을 변경하기 이전의 분포 (오른쪽) mAP_s, mAP_m, mAP_L의 기준을 변경한 이후의 분포

- Size 별 Bounding Box의 개수의 Imbalance 문제
  - bbox의 사이즈별로 count 했을 경우 Large 사이즈가 대부분을 차지
  - 기존 (512, 512) 사이즈의 이미지를 이번 대회에서 (1024, 1024)로 upscaling 시키며 bbox의 크기도 4배가 되어 대부분 Large box 범위에 들어가 imbalance 발생
  - AP 연산으로 계산되는 mAP_s, mAP_m, mAP_L이 제대로된 정보를 전달하기 힘들다고 판단하여 mmdetection evaluate에서 small, middle 범위를 4배씩 늘리도록 코드를 수정
  - 여전히 낮은 mAP_s 수치를 보여 small object 예측 문제 해결이 성능에 큰 영향을 줄 것이라 추측
  - mAP small : 0.01 → 0.05, mAP middle : 0.1 → 0.2, mAP large : 0.6 (변화없음)

|오분류된 데이터 확인 작업 수행

- fiftyone 를 활용하여 데이터를 시각화 한 후, 팀원들과 train data에 대해 전수조사 진행
- 데이터 클렌징이 필요한 데이터는 구글 스프레드시트에 표기하였고, 이후 팀원 간의 협의를 통해 최종적으로 191장을 선별 및 label 수정을 진행

태그	내용
1	담배꽁초 인식 X
2	paper/styrofoam/plastic bag/plastic에 부착된 테이프 관련
3	BBox 분류가 잘못됨
4	투명 비닐봉투 안에 있는 쓰레기 인식 관련
5	병뚜껑과 병의 재질이 다름
6	쓰레기가 없는 영역에 바운딩 박스 표시
7	쓰레기가 있는 영역에 바운딩 박스 미표시
8	영역이 애매함

- 특정 항목에 대해서 일관성 없는 Labeling 문제
  - 병뚜껑 Labeling 유무
  - 투명 비닐봉투 내부 쓰레기 Labeling 유무
  - 명함, 목장갑, 삼선슬리퍼 등 Labeling 미통일
- Miss Labeling
  - 약 20가지의 잘못된 Labeling이 존재

C. Project Management & Cooperation Tool

협업 툴
- 코드 정리 → **Github**
- 실험 내용 + 제출 내용 정리 → **Notion**
- 실험 관리 → **wandb / mlflow**

개발 환경
- 에디터 → **pycharm / VS code**

04. Result

A. Data

- Class별 imbalance는 mAP에 큰 영향이 없는것을 확인하였기 때문에 Size 간 Imbalance와 Data Cleansing에 초점을 맞추어 전처리 진행
- Data Cleansing
  - coco annotator 를 활용하여 bounding box의 영역을 수정하거나, category를 변경하는 등 데이터 클렌징 작업을 진행
  - 데이터 클렌징이 필요하다고 제안된 모든 데이터 191장를 수정하는 것보다, 명백한 오류가 있는 데이터(ex. 카테고리가 잘못 분류된 경우) 17장만 수정하는 것으로 결정
- Augmentation
  - Simple Data Augmentation
    - 각 transform을 단일로 사용할 때보다 누적해서 적용할 경우 성능 향상에 더 도움이 되었음
      - VerticalFlip / HorizontalFlip / GaussNoise / GassianBlur / Blur / ShiftScaleRotate / RandomRotate90
  - ColorJitter
    - Swin Transformer backbone + Cascade R-CNN detector에서는 성능 향상에 도움이 되어서 적용하기로 결정
    - YoloV5에서는 성능 향상에 도움이 되지 않아서 적용하지 않기로 결정
  - Oversampling
    - patch augmentation을 활용하여 train data의 oversampling 진행



B. Loss

- CV값은 비슷하나 LB값은 CIoU가 가장 적은 차이를 보임
- GIoU와 CIoU는 꾸준한 우상향 그래프를 그렸는데 DIoU는 안정되지 못하게 지그재그를 그리면서 상승
- Swin Transformer backbone + Cascade R-CNN 외의 모델은 자체 설정된 loss를 사용

	CV	LB
GIoU	0.614	0.6090
DIoU	0.615	0.6084
CIoU	0.616	0.6144

C. Model

- neck/nms/augmenation/backbone model/detector model 등을 변경하여 모델 성능 실험
- mAP 50 등을 고려해서 Ensemble을 진행할 모델 선정
- Cascade R-CNN + Swin Transformer Large 224, Cascade R-CNN + Swin Transformer Large 384, YoloV5 , YoloR, EfficientDet_d4_ap
- 최종 선택한 모델에 대하여 Weighted Box Fusion으로 Ensemble 진행

D. Hyperparameter Tuning

- Neck 실험
  - NASFCOS-FPN, PA-FPN, RFP, FPN
- NMS 실험
  - NMS, Soft-NMS
- Anchor box 실험
  - anchor scale, anchor ratio
- Optimizer 실험 : AdamW, SGD
- YoloV5 Batch size 실험 : 4, 6

E. Ensemble

- Test Time Augmentation : Multi scale, Flip
- Weighted Box Fusion : IoU 0.5 / 0.55 / 0.6 / 0.65
- Tile Ensemble

05. Evaluation

A. Opinion

- Strong

1. LB 0.7221 (최종 4위)
2. Notion / GitHub를 적극적으로 활용하여 협업하여 팀원 간의 활발한 의사소통
3. 실험 진행 중 발생한 에러 등을 즉각적으로 공유하여 개선
4. 팀원이 각자 맡은 바 역할을 책임감 있게 진행

- Weak

1. 한정된 시간을 효율적으로 사용하기 위해서는 모델 사이즈만을 줄이는 것뿐만 아니라 데이터 사이즈도 줄이는 등 통제변인을 다양하게 설정하여 시간을 잘 활용할 수 있었으면 더 다양한 실험이 가능했을 것이라고 생각함
2. 초반부터 많은 커뮤니케이션을 통하여 다양한 실험을 진행 하였으나 중반이 넘어가면서 시간적인 압박같은 환경적인 이유로 인하여 커뮤니케이션에 조금 소홀해진 것을 느껴 다음 프로젝트 부터는 다양한 툴와 룰을 이용하여 보완하고자 함
3. 프로젝트 중반부터 시도하여 완벽하게 구현하지 못한 아이디어들이 다소 존재하여 아쉬움

개인 회고 : 김나영_T3020

학습 과정 + 모델 개선을 위해 노력한 점

- 3주의 대회 기간동안 배운 내용, 에러 발생 원인 및 대처 방법, 실험해야 할 내용(시도해볼 내용) 등을 깃허브 private 레포지토리에 업로드했다. 지난 대회에서는 노션 페이지에만 기록했는데, 이번 대회에서는 깃허브에 익숙해지기 위해서 vscode로 커밋하면서 연습했다.
- 환경 설정을 할 때에는 최대한 자세하게 노션에 기록해두는 것이 나중에 관련 실험을 재현할 때 큰 도움이 된다는 것을 느꼈다.
- Swin Transformer backbone + Cascade R-CNN 모델 실험을 위주로 진행했다. ColorJitter augmentation / TTA시, Multi Scale 적용 / K-fold / Anchor Box (ratio, scale 변경) / Oversampling 적용
    - EfficientDet의 augmentation 진행하기 위해서 effdet에 이미 존재하던 transforms.py 파일에서 코드를 수정했는데 에러가 발생했고, 이를 해결하는 데에 많은 시간을 소요했다. (노션)
    - 에러를 해결하는 과정에서 transform.py의 클래스 구조를 이해할 수 있었고, 그 구조에 맞게 ColorJitter 클래스를 구현해서 에러를 해결했다.

배운 점

- 시도하고 싶은 것을 지체없이 진행하기 위해서는 대회를 시작하기 전에 미리 여러 모델, 툴 등에 대해 사전조사가 필요하다고 느꼈다.
- EfficientDet은 TTA로 Multi scale을 활용할 수 없다.
- EfficientNet을 backbone으로 사용하고 있기 때문에 input size를 고정시켜야 하므로, TTA로 Multi Scale을 적용할 수 없다.
- 개별 모델의 성능이 좋다고 해서 앙상블 했을 때, 반드시 좋은 성능으로 이어지는 것은 아니다. (ex. 가장 높은 mAP를 보인 yolov5 모델로 앙상블했을 때보다 덜 높은 mAP를 보인 yolov5 모델로 앙상블했을 때 결과가 더 좋은 경우도 있었다.)
- 같은 Augmentation 이더라도 다른 모델에 대해서는 성능이 다르게 나타날 수 있다.
  (ex. Swin Transformer + Cascade R-CNN에서는 ColorJitter를 적용하는 것이 성능을 크게 향상시켰으나,
  yolov5에서는 ColorJitter를 적용하는 것이 성능 하락으로 이어짐)

시도한 것

- boxinst를 활용해서 segmentation 정보가 필요한 Hybrid Task Cascade 모델을 학습시키고 싶었다.
    - HTC 논문을 읽고, boxinst 깃허브를 참고해서 환경설정 후 실행까지 했지만, 시간이 부족해서 더 이상 진행하지 못했다. 앞으로 시도해보고 싶다.
- 2D보다 더 많은 정보를 활용해서 task를 진행하는 3D object detection을 이번 대회에 적용해볼 수 있는지 궁금해서 라이브러리를 찾아보고 깃허브를 참고해서 환경설정 및 설치를 진행했다.
    - ‘segmentation 정보를 만들어주는 boxinst처럼, 2D dataset을 3D dataset으로 변환하는 툴이 존재하지 않을까?’ 생각이 들어 조사했지만, 대회 기간 내에 찾지 못했다. 앞으로 시도해보고 싶다.

한계 / 아쉬웠던 점

- YoloR과 EfficientDet은 모델의 여러 부분을 수정해보지 못하고 기본 config상태에서만 실험을 하다가 종료한 것이 아쉽다. 보다 빨리 시도했더라면 좋았을 것 같다.
- EDA를 형식적으로만 진행한 것 같아서 아쉬웠다. (ex. 카테고리별 bbox 분포) 데이터셋에서 발견할 수 있는 인사이트를 잘 찾아내지 못해서 아쉬웠다. 다음에는 EDA에서 시간을 더 들여서 라도 충분히 여러 가설을 생각해본 뒤에 실험을 진행해야 겠다.
- WBF 논문을 읽고, 코드를 공부하고, 관련 라이브러리를 찾는 데에 그친 것이 아쉽다. 활용해서 결과까지 확인해보고 싶다.

시도해보고 싶은 것

- 멘토님께서 의견을 주셨던 것처럼, fast api를 공부해서 object detection에서 실험한 모델을 웹사이트에 적용해보고 싶다.
- detectron2 등 다른 object detection 라이브러리도 사용해보고 싶다.

개인 회고 : 신규범_T3116

프로젝트를 통한 학습 목표

이번 프로젝트에서는 모델 중심적인 방법론이 아닌 그 외의 방법으로 의미 있는 결과를 얻는 것을 목표로 삼아 다양한 Augmentation과 Inference 방법을 중심적으로 조사하고 시행해 보았다.

프로젝트에서 진행한 부분

1. BBox가 겹쳐 있는 이미지가 상당히 많이 존재하였고, Data의 Imbalance도 크게 존재하였기 때문에 Task의 난이도가 높다고 추측되어 성능을 최우선으로 하는 2 Stage + Swin Transformer를 메인 모델로 선정하였다. Large Model은 시간적 제한이 존재하는 실험에 적합하지 않기 때문에 Small Model로 실험을 진행한 뒤 Large Model에 적용하는 방법을 주장, 이를 통하여 많은 실험이 가능했다
2. Weight Box Fusion을 통하여 Ensemble을 진행하였고, 다양한 Parameter로 실험하면서 최적의 IoU를 얻었다
3. Upscaling이 이루어진 Image라는 부분에서 현재 Metric의 타당성에 대해서 의문을 제기하고, 더 나은 평가 방법을 위하여 pycocotools Module의 코드를 Task에 맞게 수정하였다
4. Inference 부분의 Hyper Parameter 조절을 통하여 LB의 변화를 테스트해보았다
   soft nms가 단일 모델로서 성능을 향상시키지만 soft nms로 뽑은 단일모델과 yolo계열의 one stage Detector와 Weight Box Fusion을 시킬 경우 오히려 성능의 하락을 불러 일으킨다는 것을 실험을 통하여 발견하였다
5. 기존 IoU loss의 대용으로 GIoU, DIoU, CIoU에 대해서 비교 실험을 진행하였고, 가장 CV와 LB가 근접한 값을 보여준 CIoU를 메인 Loss로 채택하였다

새롭게 시도했던 부분

1. 이미지를 나누어서 따로 Inference 한 뒤 합쳐서 Inference를 진행하는 tiling 기법을 시도해 보았지만, 합치는 과정에서 많은 heuristics한 Parameter Setting이 필요한 것을 인지하였고, 현재의 방식으로 제한적인 제출 회수가 크게 소모될 것으로 예상되어 더 진행하지 못하였다. 추후 직접 구현을 위하여 크기에 따른 Bbox filter와 위치에 따른 filter를 구현해 볼 예정이다
   논문 링크
2. OpenCV를 이용하여 BBox를 Crop함과 동시에 배경에 대한 Mask Data를 생성하고, Albumentation을 이용한 Image와 Mask 회전, Random한 좌표에 붙여넣기, IoU 연산을 통한 BBox가 겹치지 않게 붙여넣기를 활용하여 Over Sampling을 위한 다양한 Image를 생성하였다. 이를 통하여 Small & Middle Object에 대한 mAP 상승을 이루어 내었다

한계 및 느낀 점

효율적인 실험을 위하여 가벼운 모델로 학습을 시켰지만 Cascade RCNN의 구조와 Data의 크기로 인하여 가장 가벼운 모델도 4~5시간의 소요되었기 때문에 Image의 사이즈도 줄여서 학습시켰으면 다양한 시도를 더 해볼 수 있지 않았을까 하는 아쉬움이 따랐다.

또한 mmdetection의 라이브러리를 다루면서 직접 모델 구조를 바꾸거나 Task를 분리하는 등의 작업이 거의 이루어지기 힘들었기 때문에 일부 Parameter를 고치는 등의 학습에 대한 피로도가 상당한 프로젝트였다. 그래서 커스텀하기 힘든 Model 쪽이 아닌 Data쪽으로 많은 시도를 해보려고 노력을 했고, 이를 실현시키기 위해 수업에서 배운 OpenCV를 이용한 이미지 변환을 사용하는 코드를 작성 해 보았기에 전체적으로 만족스러운 Project가 아니었나 생각이 든다.

이 외에도 Object만 탐지하는 모델 + Classfication 모델로 분리시킨 모델 구조 등도 생각을 해보았으나 시간적인 제한으로 인하여 실현하지 못해서 추후의 과제로 시도해보고자 한다.

개인 회고 : 이정수_T3163

PART1. 목표 설정

이론적인 이해를 기반으로 여러 가설을 설정하고 실험을 하고자 하였다. 변인 통제하여 여러 실험을 진행하여 효과적인 기법이나 모델을 파악하고자 하였다. 각 기법이나 모델 선택에 대해 논문 등 적절한 판단 근거에 따라 적용하고자 하였다.

PART2. 수행 내용 및 과정

I. 데이터 re라벨링

데이터의 bbox와 라벨링이 잘못 되어있는 부분이 있음을 확인하였고, 팀 전체적으로 라벨링 전수조사를 실시하였다. 확실히 잘못 라벨링 된 것 들을 수정한 경우, 병의 뚜껑을 인식하지 않도록 한 경우, 그리고 본인이 실험한 병의 뚜껑을 모두 인식 하도록 한 경우 모두 성능 향상이 없었다. 뚜껑을 잡아주는 게 성능이 오를 것 이라 생각했으나 test 정답 값 자체도 따로 안 잡는 것을 정답으로 둔 것 같다. 이 부분은 제공된 train데이터 에서도 대부분 안 잡았고 잡은 것들이 매우 드물었는데, 여기선 잡은 것을 노이즈로 보고 있다는 것을 실험을 하고 나서야 파악했다.

II. NMS

NMS 부분의 변경에 따른 실험을 맡아서, 이론을 찾아보고 실험을 진행하였다. 처음에는 Train 부분만 NMS 0.6 / Soft NMS 0.3, 0.4, 0.5로 나누어 실험해보며 성능을 확인해보았고 규범님이 Test 부분을 바꾸는 게 효과가 크다고 해서 Test 부분도 NMS 0.5 / Soft NMS 0.5 로 진행해보았다. 결론적으로는 단일 모델에서는 train : Soft NMS 0.5 & test : Soft NMS 0.5 조합이 가장 성능이 좋았다.
그러나 앙상블에서 Soft NMS를 적용한 경우는 성능이 떨어짐을 확인하였고, train과 test에서 모두 NMS를 적용하였다.
이후 피어세션에서 논의를 통해 그 이유가 WBF 과정에서 융합하는 과정에서 score를 averag로 계산하는데 Soft NMS가 적용된 경우 bbox가 겹쳐 있으면 뒤에 있는 bbox의 score값이 낮아지기 때문에 추후 average로 WBF을 이용해서 앙상블 할 때 성능이 떨어지는 것 같다는 결론을 얻었다.

III. EfficientDet

본 조는 2주에 한번씩 논문 발표를 진행하기로 했다. 본인의 발표 순서가 되어서, 대회와 관련 있는 논문을 읽고자 하였고, 1 Stage Detector 모델인 EfficientDet에 대해 공부 후 발표를 진행하였다. 그 후 속도가 빠르다는 장점과 근 2년내 Object Detection Competition에서 좋은 성적을 낸 적이 있음을 확인하고 이번 대회에도 써보고자 하였다. 메모리 문제로 최신 모델은 쓰지 못했고, Efficientdet d4-ap 모델을 사용하였다.

시행착오1) 당시엔 Soft NMS의 경우 앙상블 결과가 잘 안 나오는 이유에 대해서 명확히 알지 못해서, 해당 모델은 다른 결과를 낼 수도 있으니 NMS와 Soft NMS를 쓴 경우 모두 실험해 보았다. 단일모델에서는 역시 Soft NMS의 경우가 성능이 더 높았고 빠르게 수렴하였다. 그 후 앙상블시Soft NMS를 적용한 경우 왜 성능이 낮은지를 알게 되어서 Soft NMS의 결과를 앙상블 적용해보진 않았다.

시행착오2) 처음 실험은 k-fold를 적용하지 않고 train데이터로 학습하고 train데이터로 검증해서 CV score(0.77)와 LB score(0.57)의 차이가 많이 났다. 그 후 k-fold를 적용하였고 추가로 oversampling 된 데이터도 추가해서 학습시켰다. 하지만 성능이 떨어졌다. 명확한 원인 파악을 위해서는 k-fold적용만 한 후 결과보고, k-fold에 oversampling 적용한 후에 다시 결과 봤어야 했는데, 대회 막바지라 시간이 부족해 제대로 진행하지 못하였다. 추후 팀원들과 논의하며 1 Stage Model이라 더 성능이 떨어진 것 같다는 의견이 나왔다.

시행착오 3) 다른 모델들에서 효과가 있었던 TTA(multi-scale)을 적용하여 성능을 올리고자 하였다. 회의때는 적용가능할거라 생각하고관련 자료들을 찾았는데, 자료가 적었지만 multi scale 이 아닌 다른TTA들을 Efficientdet에 적용한 자료 몇 개를 찾아서 이를 참고하여 코드를 수정하였다. 에러가 너무 많이 나서 도움을 받고자 멘토님께 질문을 드렸고, 논의 과정에서 EfficientDet이 1 Stage Detector라 multi scale TTA가 적용이 안됨을 알게 되었다.

시행착오 4) 멘토링 과정에서 멘토님이 시도해보라고 추천 해주셨던 grid mask와 비슷한 Augmentation인 'CoarseDropout'을 albumentaions 라이브러리에서 제공하여 적용해보았다. CV score가 너무 낮고 0.4대 초반에 머무르고 그 이상을 오르지 않아서 LB 제출을 하지않았는데, 'CoarseDropout'를 제외하고 다시 실험한 결과를 제출한 경우 CV score가 0.44였는데 제출하니 LB 결과가 0.54였다. 그래서 'CoarseDropout' 적용한 결과의 LB score를 보지 못한 게 아쉬움이 남는다.
EfficientDet 실험을 진행하면서 자료들을 더 찾아보고 실험을 했으면 실패를 줄일 수 있었을 거라는 생각이 든다. 결론적으로는 여러 실험을 했지만 결과가 좋지 않아서 train data 전체로 학습하여 단일모델 0.57의 결과를 낸 것을 앙상블에 추가하였고LB score가 0.7220 -> 0.7221로 미세하게 성능이 올랐다.

PART3. 아쉬운 점이나 개선할 점

Object Detection Task가 처음이라 전반적인 이해가 부족했으나 그에 비해 추가적인 노력이 부족했던 것 같다. 부족한 만큼 더 공부를 진행할 것이고, 관련 실험을 진행한 자료들을 많이 찾아보고 어떤 실험을 진행할지 계획을 세워야 되겠다. 또한 모른다는 것을 부끄러워하지않고 인정하고 질문하며 공부할 것 이다.

개인 회고 : 이현홍_T3175

○ 나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?

■ 우리 팀과 나의 학습목표는 무엇이었나?

개인적으로는 Object Detection의 작동 원리와 절차를 경험해보는 자체가 목표였습니다.
지난 대회에서는 학습 절차가 어떻게 되는지 알게 되었다면, 이번 대회에서는 Object Detection을 위해
어떤 부분에 중점을 두어야 하는 지 알아보고자 하였고, 팀적으로도 이에 대한 폭넓은 이해를 하는 것이 목표였습니다.

■ 개인 학습 측면

앙상블에 활용할 수 있는 다양한 모델을 팀적으로 테스트해보는 과정에서
CNN 계열 모델인 DetectoRS ResNeXt를 담당하여 코드를 작성해보고 학습을 진행시켰습니다.

■ 공동 학습 측면

Data Cleansing을 함께 진행하였고, 상대적으로 학습 속도가 빨랐던 CNN 계열 모델의 장점을 바탕으로 적용할
Data Augmentation 기법을 찾아보고자 하였습니다.

○ 나는 어떤 방식으로 모델을 개선했는가?

벤치마크 상에서 순위가 높은 사례 중, 다른 backbone을 사용하는 모델을 앙상블에 활용할 수 있도록 실험해보았고,
다양한Data Augmentation을 실험해보며 적용할 경우를 찾아보거나 Neck 등을 변경해보았습니다.

○ 내가 한 행동의 결과로 어떤 지점을 달성하고, 어떠한 깨달음을 얻었는가?

생각보다 Data Augmentation의 결과가 예상과 비슷하게 나오는 것을 보며,
사전에 생각했던 내용과 크게 다르지 않음을 확인했습니다.

○ 전과 비교해서, 내가 새롭게 시도한 변화는 무엇이고, 어떤 효과가 있었는가?

저번 대회가 완전히 처음, 그리고 이번이 두 번째 해본 대회라 걱정스러웠지만
지난 경험덕에 어떤 프로세스로 접근할 지 구상은 할 수 있었습니다.
조금 더 빠르게 실험에 뛰어들 수 있었고, 궁금한 부분에 대해 변경을 해보며
성능이 왜 변하는 지 알아볼 수 있었습니다.

○ 마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

우선 시간적으로 더 많은 실험을 하지 못한 한계가 있었습니다.
이에 대해서 사전에 인지를 하고 있었기 때문에 실험을 해볼 것과 하지 않아도 될 것을
충분히 구상 후 시작했으면 시간을 아낄 수 있었을 텐데, 예측이 어려웠습니다.
또한 아직 이론적으로도 경험적으로도 부족하다는 것을 많이 느꼈습니다.
실험은 많이 해봤으나 성능 향상을 못 하거나 시간 효율이 나쁜 경우가 많아
많은 향상을 이끌어낼 수 없었습니다.

○ 한계/교훈을 바탕으로 다음 프로젝트에서 스스로 새롭게 시도해볼 것은 무엇일까?

베이스라인 코드에 대해서 조금 더 확실하게 알아보고 시작하려고 합니다.
그리고 이번에 제공된 서버에 설치된 툴을 사용해보며 더 최신 버전의 툴이 존재하고,
더 많은 기능을 제공하는 것을 알게 되었습니다.
기능들이 대부분 우리가 실험해볼 만한 기능이었기 때문에, 서버를 제공받으면
제공된 툴에 대해서도 확인해보면 좋을 것 같습니다.
데이터로부터 학습할 요소를 미리 예상해보고 가설을 세워 해당 부분을 더욱 강하게 학습할 수 있도록
다른 부분의 영향을 줄일 수 있는 Augmentation을 구상해보겠습니다.

개인 회고: 전수민_T3191

이번 프로젝트에서 나의 목표는 무엇이었는가?

Object Detection 프로젝트를 처음 진행해보았기에, 먼저 배운 이론들을 실제로 적용해보는 것과, 처음부터 끝까지의 프로세스를 이해하고 진행하는 것을 목표로 정했다. 다음으로는, 이전 대회 프로젝트에서 아쉬운 부분으로 남았던 부분인, 체계적인 실험관리와 효율적인 협업을 진행하는 것을 목표로 삼고 프로젝트에 임했다.

나는 내 학습 목표를 달성하기 위해 무엇을 어떻게 했는가?

정해진 기간이 있기에, 대회를 진행하면서도 학습을 잘 해나가기 위한 계획을 수립하였다.
모델을 활용하기 위한 라이브러리 사용법과 코드 활용법을 먼저 학습하고 실험을 해나가면서, 이론적인 부분은 이후에 논문을 보고 이해하며 간극을 메워나갔다. 예로 Swin Transformer 의 경우, 코드로 이해가지 않았던 부분이 논문을 보고 이해가 되고, 논문으로 이해되지 않는 부분이 코드를 통해 이해가 되며 더 깊은 이해를 할 수 있었다. 실험 관리와 협업에 있어서는 wandb 를 통해 체계적으로 실험을 구분하고 결과를 비교하였으며, Notion, Github를 이용하여 팀원간에 내용을 공유하고 피드백을 주고 받았다.

나는 어떠한 방식으로 모델을 개선했는가?

가장 먼저, 통제 변인을 명확하게 설정하기 위해 기준이 되는 Model 을 학습시키고 성능을 기록해두었다.
이후, EDA 를 통해 나온 아이디어를 기반으로, Data Cleansing, Data Augmentation, TTA, Hyperparameter Tuning 을 진행하며 단일 모델의 성능을 개선해나갔다. TTA시에 mult-scale 부분과 flip 부분을 수정 적용하여 성능을 향상시켰다.
앙상블을 고려하여 다른 계열의 모델(YoloV5)을 학습시키고 추가하는 방식으로도 성능을 개선하였다.
기존 Swin Model 에 1 Stage 계열인 YoloV5 를 추가로 앙상블하였을 때에 성능이 개선되었다.

내가 한 행동의 결과로 어떤 지점을 달성하고, 어떠한 깨달음을 얻었는가?

여러 실험을 진행하며 성능이 개선된 실험과 개선되지 않은 실험이 존재하였다. 대회 데이터 기준, TTA 시에 Multi-scale 부분을 수정 적용한 실험에서는 성능이 향상되었다. Yolo 계열 모델을 추가하여 앙상블을 진행한 경우에도 성능이 향상되었다. Swin Model 은 CIoU Loss 를 사용하였을 때, YoloV5 에서는 SGD Optimizer 를 사용하였을 때 더 나은 성능을 갖는 것을 비교 실험을 통해 보였다. 반대로, Data Cleansing, Data Augmentation 추가 적용, batch size 와 image size 변경 등의 실험에서는 성능 하락이 발생하였다.
이러한 과정 속에서, 특정 가설을 세운 이후 실제 실험을 하여 결과를 확인하는 과정의 중요성을 느낄 수 있었다. EDA를 통해 합리적이라고 생각하는 가설을 세우더라도 성능은 하락하는 경우가 많았다. 그리고 YoloV5 모델과 같은 Open Source 모델을 사용할 때에 default 값을 많이 변경하면 성능이 하락할 가능성이 높다는 것도 알게 되었다. 모델 설계자가 최적화된 augmentation, hyperparameter 를 설정해두었을 가능성이 높기에, 해당 부분을 수정할 때에는 신중 해야 한다는 것을 깨닫게 되었다.

전과 비교해서, 내가 새롭게 시도한 변화는 무엇이고, 어떤 효과가 있었는가?

YoloV5 모델 성능 개선을 위해, TTA 시에 Multi-scale 부분을 수정 적용하여 실험을 진행하였다.
기존 TTA Code 에는 이미지 사이즈를 줄이는 방식의 scaling 만 존재하였으나,
학습 시의 Multi-scale 에서는 0.5 ~ 1.5 범위의 scaling 을 진행한다는 것을 반영하여,
이미지의 크기를 키우는 scaling 을 추가하였다. EDA 를 통해 파악한 trash 데이터 특성상,
같은 class 내에서도 다양한 bbox 크기가 존재한다는 것도 원인으로 작용하였다.
해당 시도를 통해 리더보드 기준 0.56 에서, 0.58 로 0.02 의 성능이 향상되었다.
단순히 수치만을 바꾸거나 기법만을 적용하기보다, 특정 원인으로부터 출발한 가설을 검증해 본 시도였다.

마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

TTA, Ensemble 등의 몇 개의 일반적인 기법을 제외한, 새로운 시도를 통한 성능 향상을 만들어내지 못한 부분이 아쉬웠다. 미흡한 시간 관리, 경험의 부족 등이 한계로 작용했다고 생각한다. 그리고, 코드를 관리할 때에, 스스로는 Github 를 더욱 활용하지 못한 부분이 아쉬웠다. 협업하는 과정에서 Github 를 사용하는 데에 익숙하지 않았었기에, 실험을 issue 를 등록하고, Pull Request 하는 과정을 바로 진행하지 않고 이후에 작성하는 경우가 있었다.

한계 / 교훈을 바탕으로 다음 프로젝트에서 스스로 새롭게 시도해볼 것

다음 프로젝트에는 EDA 를 초반에만 진행하는 것이 아니라, 이후에도 지속적으로 진행하며 데이터에 대한 이해도를 더 높이고 다양한 가설을 실험해 볼 예정이다. 데이터를 더 다방면으로 파악할수록 다양한 가설이 나올 수 있다고 생각하기 때문이다. 그리고, Github 를 더 활발하게 활용하며 코드 리뷰를 진행하고 더 효율적인 협업을 해나갈 예정이다.